PyQT e matplotlib: un'introduzione

Tommaso Mazzoni

Sommario:

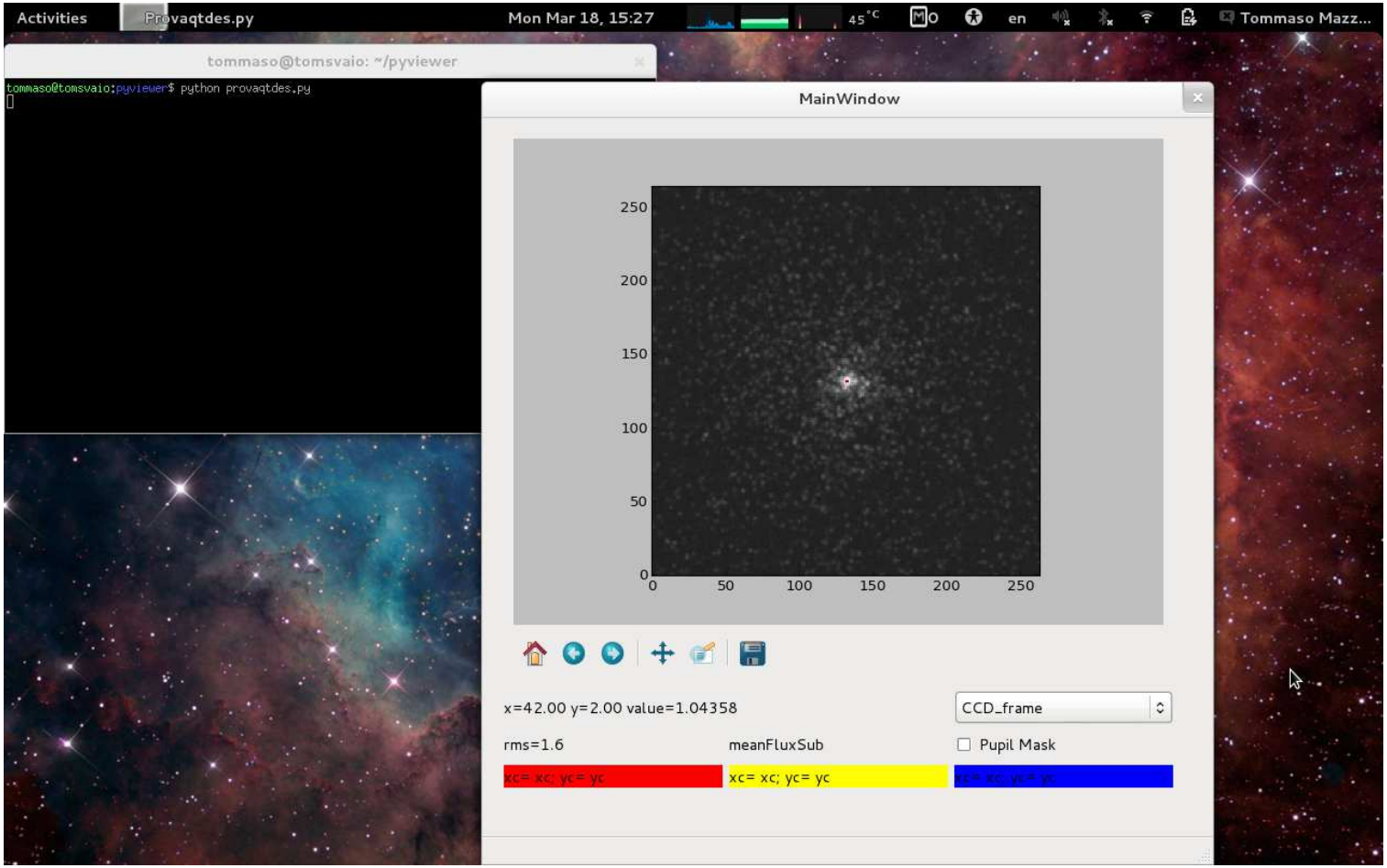
Sommario:

Sommario:

Sommario:

Sommario:

Realizzare una finestra con pyQT contenente un plot realizzato con matplotlib, ad esempio un visore per una camera CCD.



Bibliografia: Matplotlib for Python Developers (a chi interessa lo posso inviare per mail...)

```
class Ui_mainWindow(object):
def setupUi(self, MainWindow):
    MainWindow.setObjectName(_fromUtf8(MainWindow))
    MainWindow.resize(648, 696)
    self.centralwidget= QtGui.QWidget(MainWindow)
    self.centralwidget.setObjectName(
                                    _fromUtf8(centralwidget))
    self.gridLayoutWidget= QtGui.QWidget(
                                          self.centralwidget)
    self.mpl= MplWidget(self.gridLayoutWidget)
    self.mpl.setObjectName(_fromUtf8(mpl))
    self.mplCombo= QtGui.QComboBox(self.gridLayoutWidget)
    self.mplCombo.setObjectName(_fromUtf8(mplCombo))
    MainWindow.setCentralWidget(self.centralwidget)
    self.retranslateUi(MainWindow)
    QtCore.QMetaObject.connectSlotsByName(MainWindow)
```

```
class Ui_mainWindow(object):
def setupUi(self, MainWindow):
    MainWindow.setObjectName(_fromUtf8(MainWindow))
    MainWindow.resize(648, 696)
    self.centralwidget= QtGui.QWidget(MainWindow)
    self.centralwidget.setObjectName(
                                    _fromUtf8(centralwidget))
    self.gridLayoutWidget= QtGui.QWidget(
                                          self.centralwidget)
    self.mpl= MplWidget(self.gridLayoutWidget)
    self.mpl.setObjectName(_fromUtf8(mpl))
    self.mplCombo= QtGui.QComboBox(self.gridLayoutWidget)
    self.mplCombo.setObjectName(_fromUtf8(mplCombo))
    MainWindow.setCentralWidget(self.centralwidget)
    self.retranslateUi(MainWindow)
    QtCore.QMetaObject.connectSlotsByName(MainWindow)
```

```
from PyQt4 import QtCore, QtGui
from mplwidget import MplWidget

    def retranslateUi(self, MainWindow):
        MainWindow.setWindowTitle(
              QtGui.QApplication.translate(
                  MainWindow, MainWindow, None,
                       QtGui.QApplication.UnicodeUTF8))
        self.coord_label.setText(
              QtGui.QApplication.translate(
                  MainWindow, TextLabel, None,
                     QtGui.QApplication.UnicodeUTF8))
```

```
from PyQt4 import QtCore, QtGui
from mplwidget import MplWidget

    def retranslateUi(self, MainWindow):
        MainWindow.setWindowTitle(
              QtGui.QApplication.translate(
                  MainWindow, MainWindow, None,
                       QtGui.QApplication.UnicodeUTF8))
        self.coord_label.setText(
              QtGui.QApplication.translate(
                  MainWindow, TextLabel, None,
                     QtGui.QApplication.UnicodeUTF8))
```

```
from matplotlib.backends.backend_qt4agg \
import NavigationToolbar2QTAgg

 class MplWidget(QtGui.QWidget):
    def __init__(self, parent = None):
       QtGui.QWidget.__init__(self, parent)

       self.canvas = MplCanvas()

       self.ntb = NavigationToolbar2QTAgg(self.canvas,parent)

       self.vbl = QtGui.QVBoxLayout()
       self.vbl.addWidget(self.canvas)
       self.vbl.addWidget(self.ntb)
       self.setLayout(self.vbl)
```

```
from matplotlib.backends.backend_qt4agg \
import NavigationToolbar2QTAgg

 class MplWidget(QtGui.QWidget):
    def __init__(self, parent = None):
        QtGui.QWidget.__init__(self, parent)

        self.canvas = MplCanvas()

        self.ntb = NavigationToolbar2QTAgg(self.canvas,parent)

        self.vbl = QtGui.QVBoxLayout()
        self.vbl.addWidget(self.canvas)
        self.vbl.addWidget(self.ntb)
        self.setLayout(self.vbl)
```

```
from matplotlib.backends.backend_qt4agg \
import FigureCanvasQTAgg

class MplCanvas(FigureCanvasQTAgg):
    def __init__(self):
        self.fig = Figure()
        self._a = self.fig.add_subplot(111)
        FigureCanvasQTAgg.__init__(self, self.fig)
        FigureCanvasQTAgg.setSizePolicy(
                                self,QtGui.QSizePolicy.Expanding,
                                QtGui.QSizePolicy.Expanding)
        FigureCanvasQTAgg.updateGeometry(self)
        self.setFocusPolicy(QtCore.Qt.StrongFocus)
        self.setFocus()
        self._producer= self.getFrame
        self._a.format_coord = self.imageReportPixel
```

```
        image= self._producer()
        self._im = self._a.imshow(image,
                                  origin='lower',
                                  interpolation='nearest')
        self._cbar = self.fig.colorbar(self._im)

#fine __init__

def getFrame(self):
        np.random.seed(int(time.time() % 1 * 1e6))
        sz=(264, 264)
        noise_level=0.01
        noise= np.random.randn(sz[0], sz[1])*noise_level
        ima=np.zeros((264,264))
        ima[102:162,102:162]=0.1
        return ima+noise
```

```
def imageReportPixel(self, x, y):
    im= self._im.get_array()
    self._xx = np.clip(int(np.round(x)),
                               0, im.shape[1]-1)
    self._yy = np.clip(int(np.round(y)),
                               0, im.shape[0]-1)
    self._vv = im[self._yy, self._xx]
    return " "
```

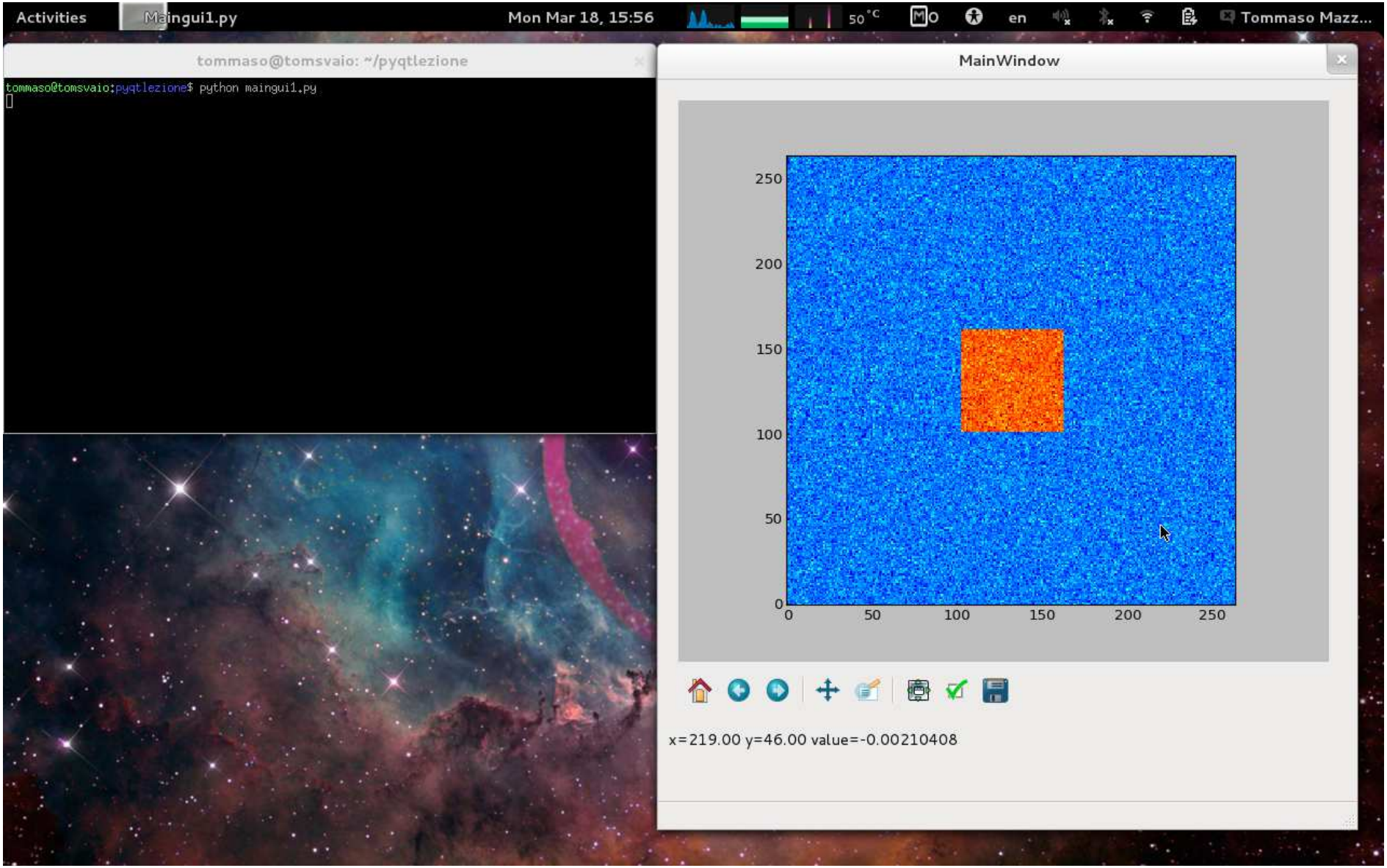
```
if __name__ == '__main__':
    # crea l'applicazione GUI
    app = QtGui.QApplication(sys.argv)
    # inizializzazione finestra
    dmw = DesignerMainWindow()
    # inizializzazione connessioni
    dmw.connection()

    dmw.show()
    # esegui
    sys.exit(app.exec_())
```

```
class DesignerMainWindow(QtGui.QMainWindow, Ui_MainWindow):
    def __init__(self, parent = None):
        super(DesignerMainWindow, self).__init__(parent)
        self.setupUi(self)

    def connection(self):
        self.cidmotion= self.mpl.canvas.mpl_connect(
                        'motion_notify_event', self.on_motion

    def on_motion(self, event):
        self.coord_xy()
    def coord_xy(self):
        x=self.mpl.canvas._xx
        y=self.mpl.canvas._yy
        z=self.mpl.canvas._vv
        self.coord_label.setText("x=%.2f y=%.2f value=%g" %
                                 (x, y, z))
```


Activities
Maingui1.py
Mon Mar 18, 15:56
50 °C
en
Tommaso Mazz...
tommaso@tomsvaio: ~/pyqtlezione
tommaso@tomsvaio:pyqtlezione$ python maingui1.py
MainWindow
250
200
150
100
50
0
0
50
100
150
200
250
x=219.00 y=46.00 value=-0.00210408

```
#in DesignerMainWindow __init__
        self._imageFuncDict={
                        'CCD_frame': self.mpl.canvas.getFrame,
                        'Dark_frame': self.darkFrame}
        self.mplCombo.addItems(self._imageFuncDict.keys())
        self.mplCombo.setCurrentIndex(0)


#in DesignerMainWindow connection
        self.mplCombo.currentIndexChanged[QtCore.QString].
                                        connect(self.setImageFunc)
        self.keypress= self.mpl.canvas.mpl_connect(
                        'key_press_event', self.key_press)

        QtCore.QObject.connect(self.mpl.canvas,
                                QtCore.SIGNAL('sigClicked'),
                                self.coord_xy)
```

```
def darkFrame(self):
    np.random.seed(int(time.time() % 1 * 1e6))
    sz=(264, 264)
    noise_level=0.01
    noise= np.random.randn(sz[0], sz[1])*noise_level
    return noise

def key_press(self, event):
    if event.key=='o':
        self.mpl.canvas._optimize_limit()
        return

def setImageFunc(self):
    self.mpl.canvas.newPlot=True
    self.mpl.canvas._producer=self._imageFuncDict[
                        str(self.mplCombo.currentText())]
```

```
#in MplCanvas __init__


        self.refreshTime=1000
        self.timer = self.startTimer(self.refreshTime)
        self.newPlot=False
```

```
def timerEvent(self, evt):
    z= self._producer()
    if self.newPlot:
        self._a.format_coord = self.imageReportPixel
        self._im = self._a.imshow(
                                z,
                                origin='lower',
                                interpolation='nearest')
        self._cbar.set_cmap(self._im.get_cmap())
        self._cbar.update_bruteforce(self._im)
        self.newPlot=False
    else:
        self._im.set_array(z)
    self._cbar.draw_all()
    self.emit(QtCore.SIGNAL("sigClicked"), ())
    self._a.format_coord(self._xx,self._yy)
    self.fig.canvas.draw()
```

```
def _optimize_limit(self):
    curr= self._im.get_array()
    std=np.std(curr)
    mean=np.mean(curr)
    std=np.std(curr)
    self._cbar.norm.vmin = mean-4*std
    self._cbar.norm.vmax = mean+4*std
    self._im.set_norm(self._cbar.norm)
    self._cbar.patch.figure.canvas.draw()
```

Il risultato? Vediamolo insieme...